Group members: Eduard Koshkelyan, Marwane Bidamou, Roshan Maharjan

Team members:

4. Bidamou, Marwane

8. Koshkelyan, Eduard

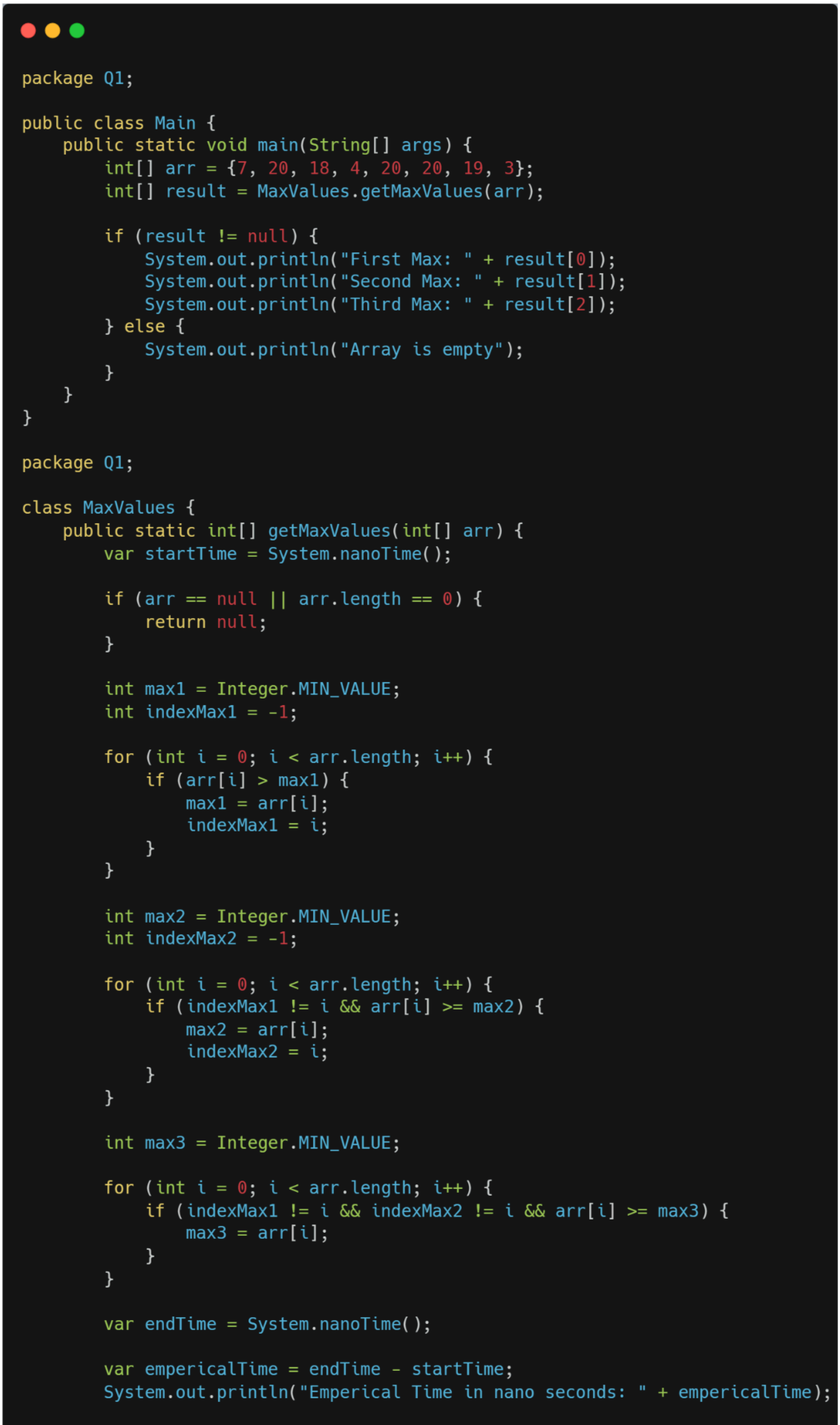
11. Maharjan, Roshan

Question 1. Comparing Algorithms. Problem: Find the THIRD largest in an array.

Algorithm 1

Idea – Use three loops one after another. First loop will find Max. Second loop will find Second Max, Third loop will find third max. Note that it is possible First max == second Max f== Third Max as in 7, 20, 18, 4, 20, 19, 20, 3. and your program should return 20 in this case

Group members: Eduard Koshkelyan, Marwane Bidamou, Roshan Maharjan



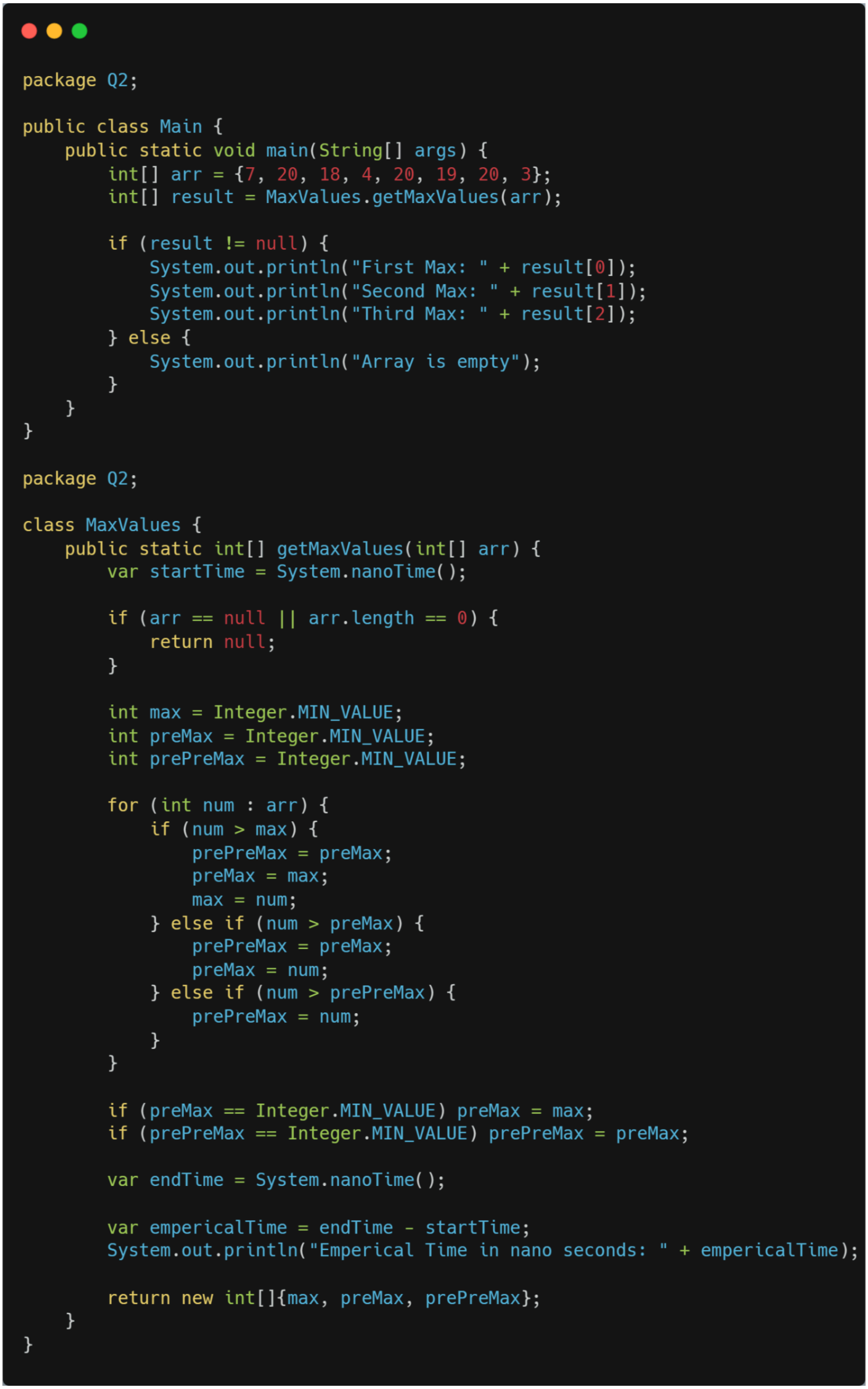
```java
package Q1;

public class Main {
    public static void main(String[] args) {
        int[] arr = {7, 20, 18, 4, 20, 20, 19, 3};
        int[] result = MaxValues.getMaxValues(arr);

        if (result != null) {
            System.out.println("First Max: " + result[0]);
            System.out.println("Second Max: " + result[1]);
            System.out.println("Third Max: " + result[2]);
        } else {
            System.out.println("Array is empty");
        }
    }
}

package Q1;

class MaxValues {
    public static int[] getMaxValues(int[] arr) {
        var startTime = System.nanoTime();

        if (arr == null || arr.length == 0) {
            return null;
        }

        int max1 = Integer.MIN_VALUE;
        int indexMax1 = -1;

        for (int i = 0; i < arr.length; i++) {
            if (arr[i] > max1) {
                max1 = arr[i];
                indexMax1 = i;
            }
        }

        int max2 = Integer.MIN_VALUE;
        int indexMax2 = -1;

        for (int i = 0; i < arr.length; i++) {
            if (indexMax1 != i && arr[i] >= max2) {
                max2 = arr[i];
                indexMax2 = i;
            }
        }

        int max3 = Integer.MIN_VALUE;

        for (int i = 0; i < arr.length; i++) {
            if (indexMax1 != i && indexMax2 != i && arr[i] >= max3) {
                max3 = arr[i];
            }
        }

        var endTime = System.nanoTime();

        var empericalTime = endTime - startTime;
        System.out.println("Emperical Time in nano seconds: " + empericalTime);
```

Group members: Eduard Koshkelyan, Marwane Bidamou, Roshan Maharjan

Algorithm 2

Idea – Use one loop. Maintain three variables max, preMax and prePreMax such that max will have the maximum value, preMax will have the second largest and prePreMax will have the third largest value.

Group members: Eduard Koshkelyan, Marwane Bidamou, Roshan Maharjan



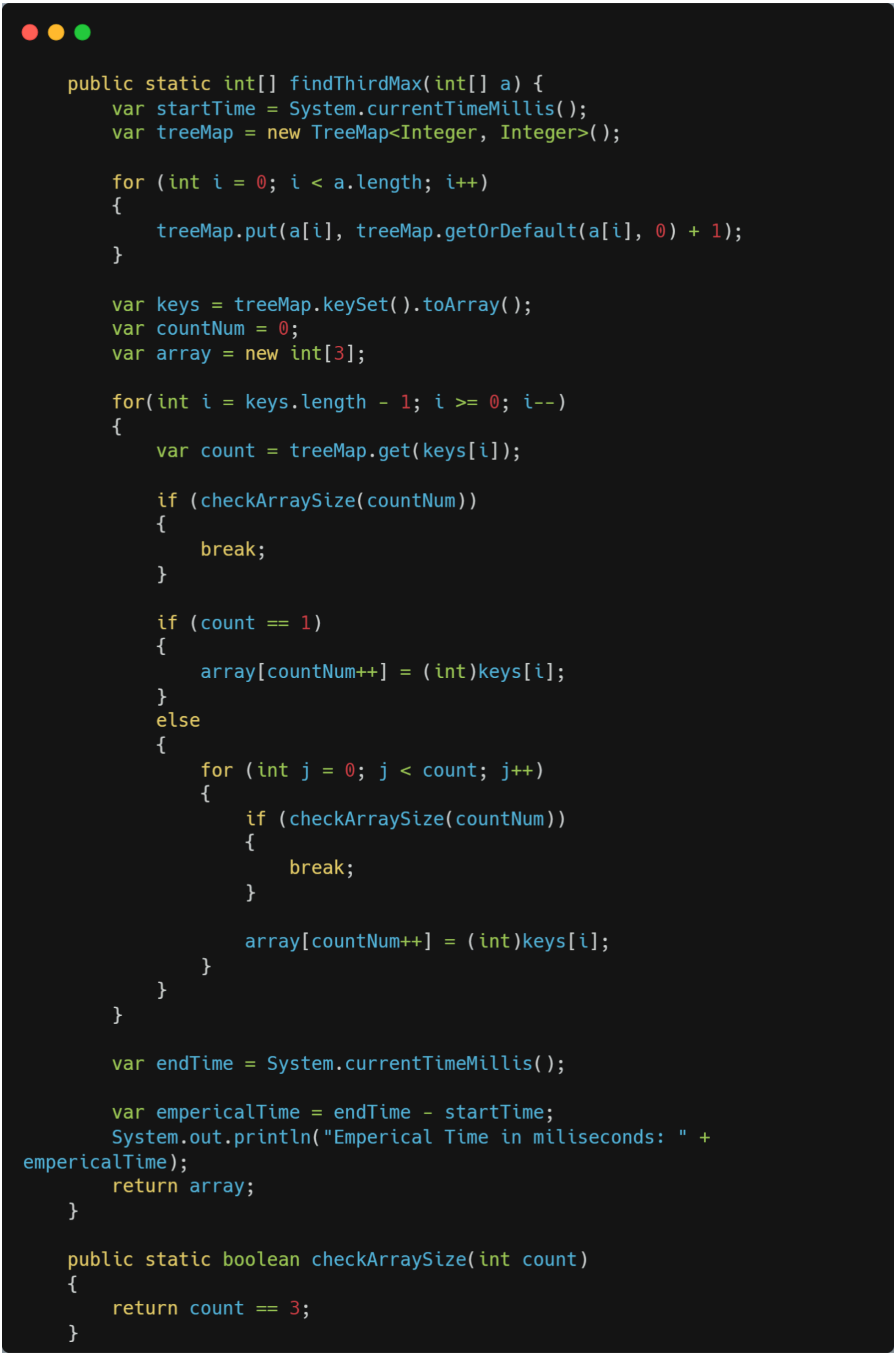
```java
package Q2;

public class Main {
    public static void main(String[] args) {
        int[] arr = {7, 20, 18, 4, 20, 19, 20, 3};
        int[] result = MaxValues.getMaxValues(arr);

        if (result != null) {
            System.out.println("First Max: " + result[0]);
            System.out.println("Second Max: " + result[1]);
            System.out.println("Third Max: " + result[2]);
        } else {
            System.out.println("Array is empty");
        }
    }
}

package Q2;

class MaxValues {
    public static int[] getMaxValues(int[] arr) {
        var startTime = System.nanoTime();

        if (arr == null || arr.length == 0) {
            return null;
        }

        int max = Integer.MIN_VALUE;
        int preMax = Integer.MIN_VALUE;
        int prePreMax = Integer.MIN_VALUE;

        for (int num : arr) {
            if (num > max) {
                prePreMax = preMax;
                preMax = max;
                max = num;
            } else if (num > preMax) {
                prePreMax = preMax;
                preMax = num;
            } else if (num > prePreMax) {
                prePreMax = num;
            }
        }

        if (preMax == Integer.MIN_VALUE) preMax = max;
        if (prePreMax == Integer.MIN_VALUE) prePreMax = preMax;

        var endTime = System.nanoTime();

        var empericalTime = endTime - startTime;
        System.out.println("Emperical Time in nano seconds: " + empericalTime);

        return new int[]{max, preMax, prePreMax};
    }
}
```

Group members: Eduard Koshkelyan, Marwane Bidamou, Roshan Maharjan

Algorithm 3

Idea – Use an ordered dictionary.

Group members: Eduard Koshkelyan, Marwane Bidamou, Roshan Maharjan



```
    public static int[] findThirdMax(int[] a) {
        var startTime = System.currentTimeMillis();
        var treeMap = new TreeMap<Integer, Integer>();

        for (int i = 0; i < a.length; i++)
        {
            treeMap.put(a[i], treeMap.getOrDefault(a[i], 0) + 1);
        }

        var keys = treeMap.keySet().toArray();
        var countNum = 0;
        var array = new int[3];

        for(int i = keys.length - 1; i >= 0; i--)
        {
            var count = treeMap.get(keys[i]);

            if (checkArraySize(countNum))
            {
                break;
            }

            if (count == 1)
            {
                array[countNum++] = (int)keys[i];
            }
            else
            {
                for (int j = 0; j < count; j++)
                {
                    if (checkArraySize(countNum))
                    {
                        break;
                    }

                    array[countNum++] = (int)keys[i];
                }
            }
        }

        var endTime = System.currentTimeMillis();

        var empericalTime = endTime - startTime;
        System.out.println("Emperical Time in miliseconds: " +
empericalTime);
        return array;
    }

    public static boolean checkArraySize(int count)
    {
        return count == 3;
    }
```

Group members: Eduard Koshkelyan, Marwane Bidamou, Roshan Maharjan

In this lab, for every algorithm you will:

- (a) write the pseudo code. (Must follow the notations and conventions used in today's Lecture)
- (b) determine the worst-case time complexity by counting as in Slide 15 Lesson 2.
- (c) Perform an empirical time comparison by implementing using Java, similar to what you did in W1D1.

Draw a chart to compare all algorithms.

(a) Pseudo Code

For Q1: Using Three Loops

```
FUNCTION getMaxValues(arr)
   IF arr is NULL OR arr is empty THEN
      RETURN NULL

   DECLARE max1 AS MIN_VALUE
   FOR EACH num IN arr DO
      IF num > max1 THEN
         max1 = num
      END IF
   END FOR

   DECLARE max2 AS MIN_VALUE
   FOR EACH num IN arr DO
      IF num >= max2 THEN
         max2 = num
      END IF
   END FOR

   DECLARE max3 AS MIN_VALUE
   FOR EACH num IN arr DO
      IF num >= max3 THEN
         max3 = num
      END IF
   END FOR

   RETURN [max1, max2, max3]
END FUNCTION
```

For Q2: Using One Loop

```
FUNCTION getMaxValues(arr)
   IF arr is NULL OR arr is empty THEN
      RETURN NULL

   DECLARE max AS MIN_VALUE
   DECLARE preMax AS MIN_VALUE
```

```
    DECLARE prePreMax AS MIN_VALUE

    FOR EACH num IN arr DO
        IF num > max THEN
            prePreMax = preMax
            preMax = max
            max = num
        ELSE IF num > preMax THEN
            prePreMax = preMax
            preMax = num
        ELSE IF num > prePreMax THEN
            prePreMax = num
        END IF
    END FOR

    IF preMax is MIN_VALUE THEN preMax = max
    IF prePreMax is MIN_VALUE THEN prePreMax = preMax

    RETURN [max, preMax, prePreMax]
END FUNCTION
```

For Q3: Using ordered dictionary

```
FUNCTION FindThirdMax(nums)
    DECLARE treeMap AS NEW TreeMap

    // Populate the treeMap with the count of each number
    FOR EACH n IN nums DO
        IF treeMap containsKey n THEN
            treeMap[n] = treeMap[n] + 1
        ELSE
            treeMap[n] = 1
        END IF
    END FOR

    DECLARE keys AS array of sorted keys from treeMap
    DECLARE countNum AS 0
    DECLARE resultArray AS new array of size 3

    // Loop through keys in descending order
    FOR i FROM length of keys - 1 DOWN TO 0 DO
        DECLARE count AS treeMap[keys[i]]

        // Stop if we've found 3 numbers
        IF countNum == 3 THEN
            BREAK
        END IF

        // If the current number appears once, add it to resultArray
        IF count == 1 THEN
            resultArray[countNum] = keys[i]
            countNum = countNum + 1
        ELSE
            // If the current number appears multiple times, add it as many times as necessary
            FOR j FROM 0 TO count - 1 DO
                IF countNum == 3 THEN
                    BREAK
                END IF
                resultArray[countNum] = keys[i]
                countNum = countNum + 1
            END FOR
        END IF
    END FOR

    RETURN resultArray
END FUNCTION
```

(b) Worst-case Time Complexity

1. For Q1:

- Each of the three loops iterates through the array once.
- Total Time Complexity: O(n + n + n) = O(3n) = O(n).

2. For Q2:

- Only a single loop iterates through the array.
- Total Time Complexity: O(n).

3. For Q3

- Populating the dictionary: O(n)
- Checking maximums: O(3)
- Total Time Complexity: O(n)

(c) Empirical Time Comparison Implementation in Java

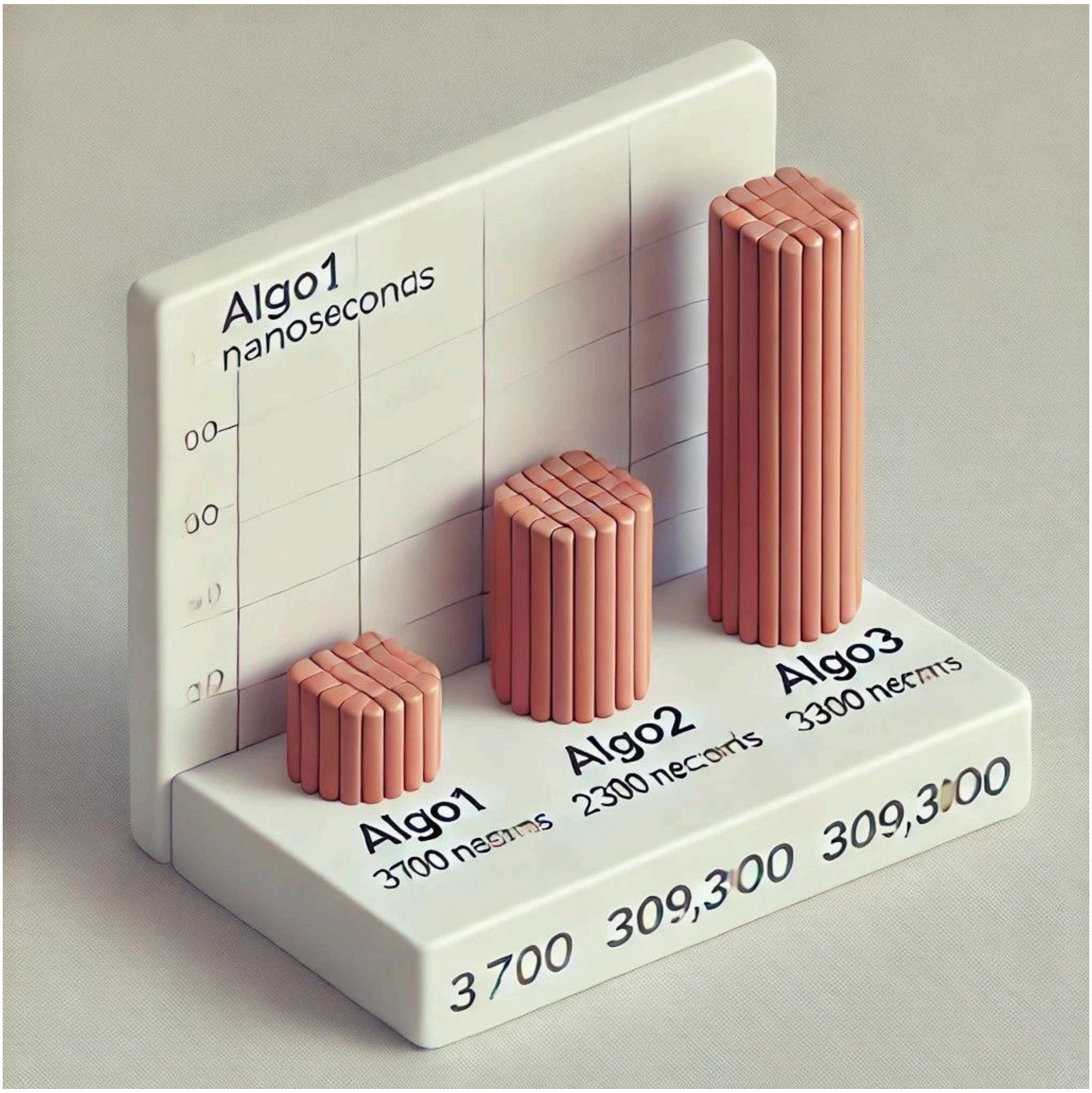
We use nano seconds as long as the operations we're performing involve simple iterations through small arrays, this might not take a noticeable amount of time in milliseconds, resulting in an empirical time of 0.

Algo1: 3700 nano seconds

Algo2: 2300 nano seconds

Algo3: 309300 nano seconds

Group members: Eduard Koshkelyan, Marwane Bidamou, Roshan Maharjan



Question 2. Consider the following functions to determine the relationships that exist among the complexity classes they belong

10, 1, n^3 , n^1/3 , log(log n), n^2 , n^1/2, logn , log n^n , n^k (k > 3), n^1/k (k > 3), nlogn, ln n, 2^n , 3^n , n^n , (n^½) logn, (n^⅓) logn, n!

Notation clarification. log(log n) = loglog n != log^2n = (log n)^2 .

The partial table is given. Your task is to complete the table. The table is in the strict ascending order. (if you have any questions, please ask.

Group members: Eduard Koshkelyan, Marwane Bidamou, Roshan Maharjan

Functions	**Growth Rate**
10, 1	O(1)
log n	O(log n)
n, ln n	O(n)
nlogn	O(nlogn)
n^⅓, $n^{1/3}$ logn	O(n^⅓)
n^½ , $n^{1/2}$ logn	O(n^½)
n 1/k (k > 3)	$O(n^{1/k})$
n^1/k (k > 3)	O(n^1/k)
n^3	O(n^3)
n^k (k > 3)	O(n^k)
2^n	$O(2^n)$
3^n	$O(3^n)$
n!	O(n!)
log n^n, n^n,	O(n^n)